Data Preprocessing → Collecting & manipulating data → Meaningful Information

Data Mining → Exraction of pattern & knowledge from large amount of data.

Data preprocessing steps   Converting a Raw Data → Clean dataset

from sklearn.preprocessing import

Rescaling → attributes with differnt scale are processed

Standardizing → the dataset   StandardScaler   MinMaxScaler   MaxAbsScaler   RobustScaler (outliers)

Normalizing → the dataset   Normalizer

Encoading → Categorical → Integer   OneHotEncoder   LableEncoder

Handiling missing → data   SimpleImputer   by defaut → stratergy = mean

Univariate

IterativeImputer → multivariate feature imputation

Removing duplicated → data points

Removing Outliers → handling noisy data

Discretize → Data   quantization or binning   contineous data → bins

kBinsDiscretizer

Split → Train-test split   skearn.model_selection → train_test_split

---

Steps in feature selection

Data Preprocessing → Feature scoring → Selection

clean & prepare data for feature sel^n

compute scores reflets importance to target value

most imp. features based on their scores

Feature engg. Vs Feature Selection

Feature Engg. → create new features — in order to help ML models

- Filling missing values → within variables ✓
- Encoading categorical variables ✓
- Variable transformation ✓

Feature Selection → from feature pool → predict target variables efficiently

Filter method

based on statistical measures & correlations

Information gain → ↓ red^n in entropy

```
from sklearn.feature_selection import mutual_info_classif
```

fisher score → algorithm returns ranks of variables w.r.t. $\boxed{}$ decending order

```
from sklearn.feature_selection import fisher_score
```

Chi-squared test → used for categorical variables

selection → based on best chi-square score

```
from sklearn.feature_selection import SelectkBest
from sklearn.feature_selection import chi2
```

Correlation Coefficients → df.corr( ) sel^n based on 0.5 threshold

drop features having lower co-efficient value.

Variance threshold → removes who doesn't meet some threshold.

removes all zero-variance features.

small drawback doesn't consider relationship b/w feature & target variable

MAD [mean absolute diffence]

similar to variance method

Dispresion Ratio → $\dfrac{\text{Arithmetic mean}}{\text{Geometric mean}}$

Wrapper method

Forword Feature Selection

forword = True

iterative method → starts with best performing features against target value

Next selects another variable gives best performance combination.

Backword Feature Elimination

forword = False

iterative method → starts with all available features against target value

Next removes one variable which improves performance combination.

```
from mlxtend.feature_selection import SequentialFeatureSelection
```

Exhaustive feature selection

most robust feature sel^n method so far. → Tries every possible combination returns best performing set.

```
from mlxtend.feature_selection import ExhaustiveFeatureSelection
```

Embedded Methods

LASSO Regularization L1

```
from sklearn.feature_selection import SelectFromModel
model = SelectFromModel ( model , prefit = True )
```

```
X_new = model.transform(X)

selected_columns = selected_features.columns [selected_features.var() != 0 ]
```

Random forest importance

model → rf

```
model.fit(X,y) → importances = model.feature_importances_
```

How we deal with missing data ?

→ * Check missing values → `df.isna().sum()`

1] Drop missing values

2] Impute values

```
from sklearn.impute import SimpleImputer
```

By deafault strategy → mean

Missing data type

MCAR → Missing Completely At Random .... ignorable
MAR → Missing At Random .... ignorable
MNAR → Missing Not At Random .... can't ignore

Pattern to missingness	Data Explains Pattern	
	yes	no
yes	MAR	MNAR
No	-	MCAR

Data Explains → other variables in dataset can explain pattern

example political poll → MAR
many people refuse to answer

1] Drop missing values

1> dropping rows where there are missing values → `df.dropna()`

2> dropping entire row / column for multiple missing values in the row

`df.dropna(axis=0 , thresh=2)` drops rows where □ > 2

3> drop entire column

`df.drop(['col_name'] , axis=1, inplace = True)`

2] Impute values

1> Replace NaN with given value

`df.col_name.fillna(0)` ← for numeric

`df.col_name.fillna('string')` ← for strings

2> Replace with mean, median, mode

`df.col_name.fillna((df.col_name.mean()))`

3> Replace with previous or next value — previous value Forword Fill 'ffill'
next value Backword Fill 'bfill'

`df.col_name.fillna(method = 'ffill')`

4> Interpolation `df.col_name.interpolate()`

Scale	Example	Description	What is meaningful ?	Central tendecy
Nominal	Name, Gender, Pin Code	Denotes name or gender. Ordering ✗	only count	Mode
Ordinal	Low, medium, high Rank → 1, 2, 3	value the orders Ordering ✓	only ordering can't measure dist. b/w values	Median Mode
Interval	Temperature °c	True zero absent	only diff. is meaningful but not the ratio	Mean median mode
Ratio Scale	Height, Weight Temperature kelvin	True zero present	Both diff. & ratio are meaningful	Mean median mode